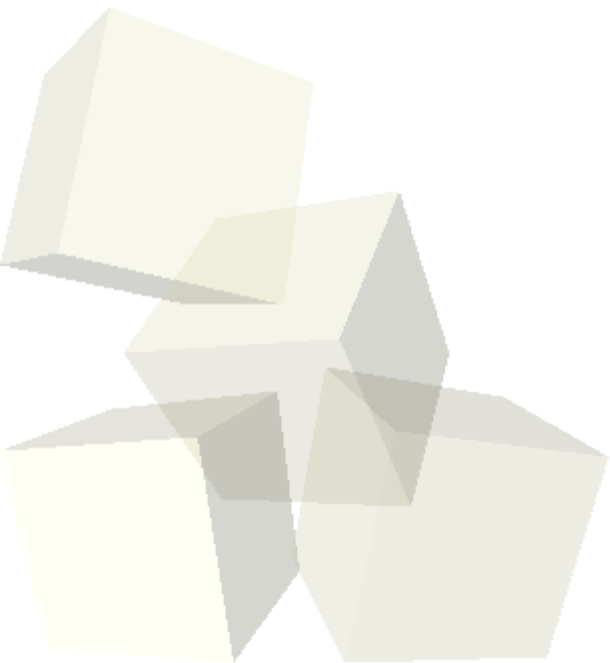




Service-Framework für das Open-Source Projekt Dotplot

Jens Peter





- Kurze Beschreibung zu Dotplot
 - ◆ Was ist Dotplot?
 - ◆ Beispiele
- Matrix4.plot
 - ◆ Eclipse-Anbindung
 - ◆ Grundlegende Funktionsweise
- Grundlegende konzeptionelle Änderungen an Matrix4.plot
 - ◆ Plotquelle
 - ◆ Typisierung von Plotquellen
 - ◆ Konfigurationsmanagement



Inhalt (2)

- Service-Framework
 - ◆ Services
 - ◆ Tasks
 - ◆ Hotspots
 - ◆ Jobs
- Plugin-Framework
 - ◆ Plugins
 - ◆ Services für das Plugin-Framework
- Services/Plugins innerhalb Matrix4.plot
- Fazit und Quellen



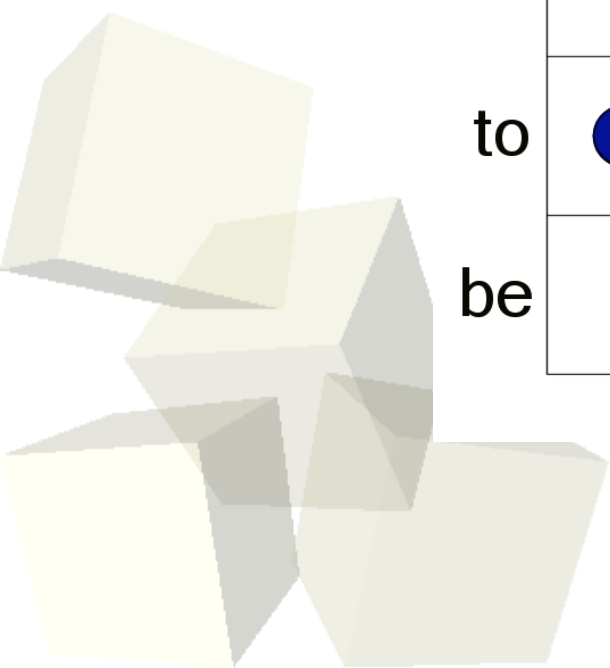
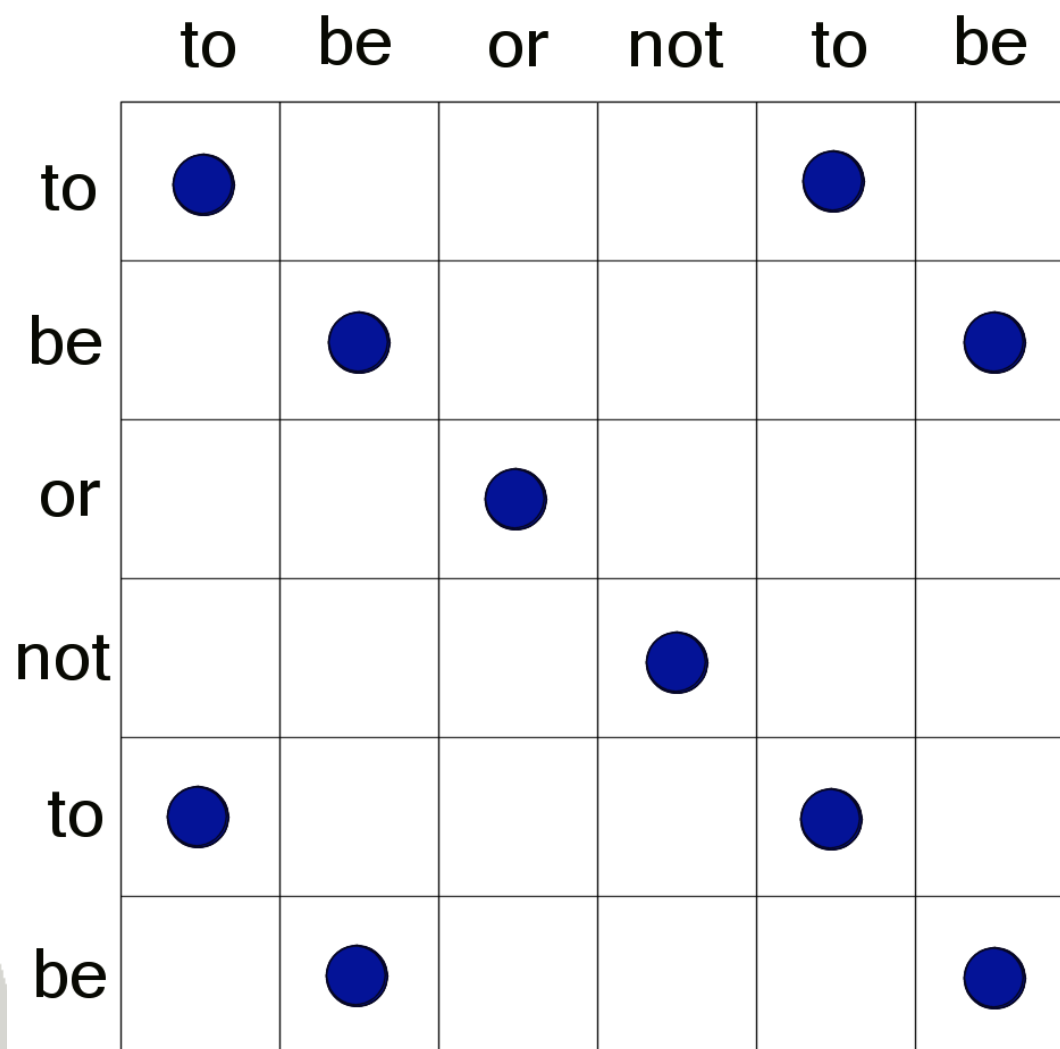
Was ist Dotplot?

- Abstammung aus der Genethik
 - ◆ Vergleich von Erbinformation
 - ◆ Graphische Visualisierung
 - ◆ Besserer Überblick
- Allgemein: Vergleich von beliebigen Anordnungen von Elementen
 - ◆ Vergleich von Tokenketten
 - ◆ Tokens:
 - z.B.: Wörter in Sprache



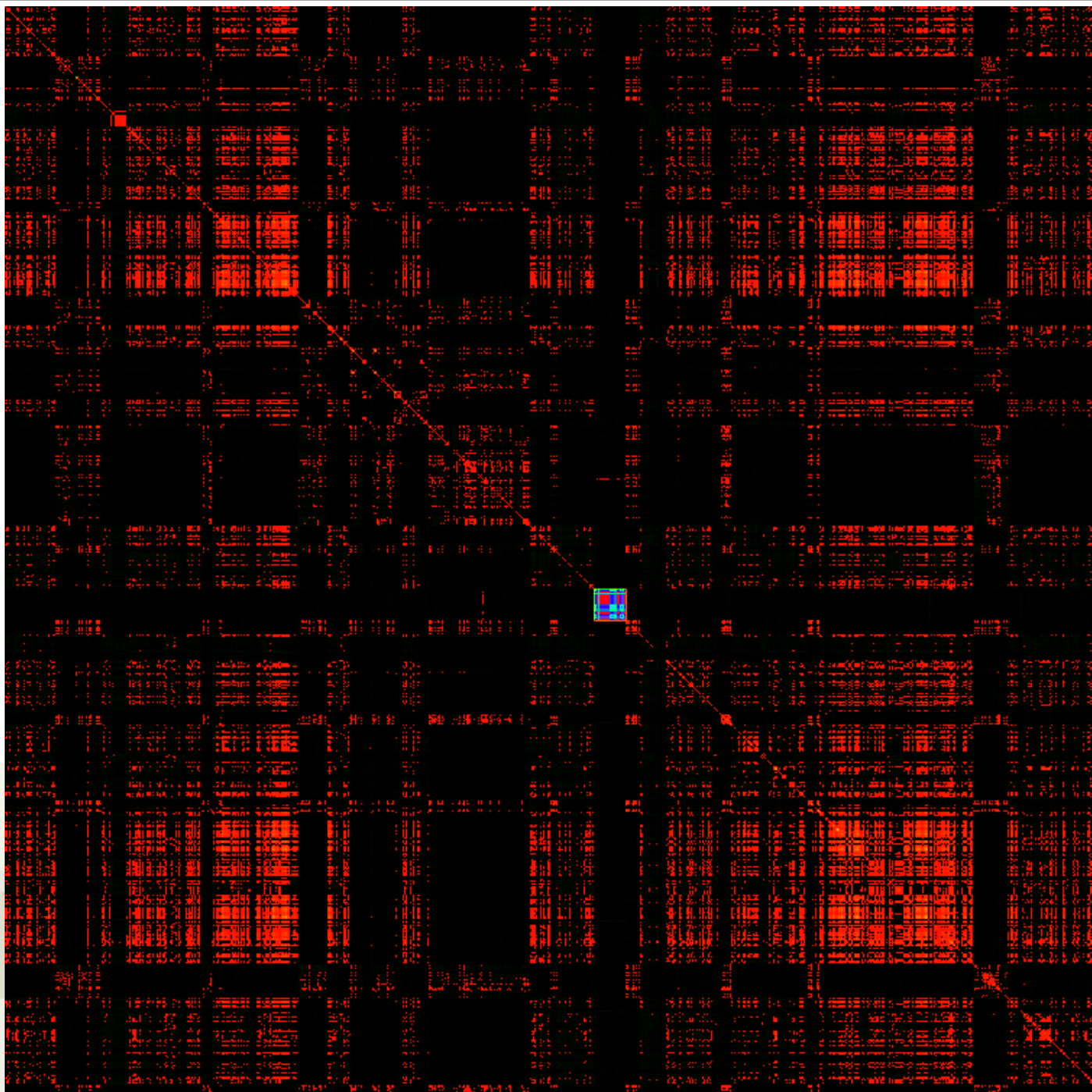


Dotplot: Beispiele





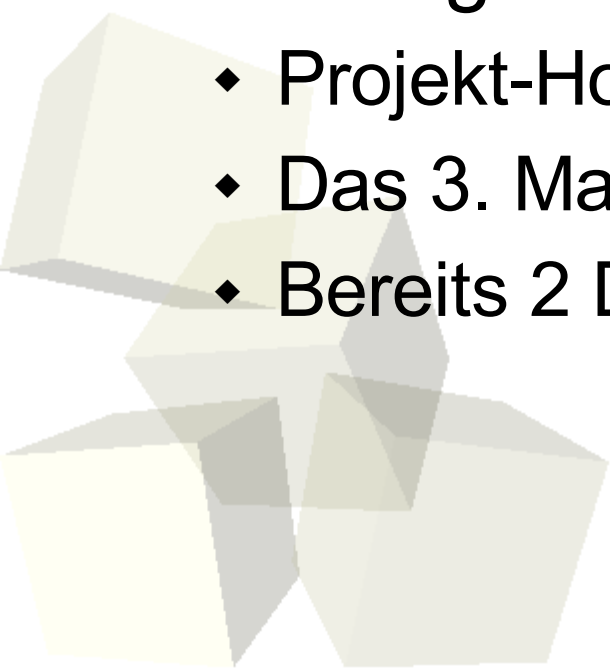
Dotplot: Beispiele





Matrix4.plot

- Projekt der FH-Giessen Friedberg
- 2004: Start innerhalb eines Schwerpunktpraktikums
- Ziel: Umsetzung der Dotplot-Idee
- Derzeitiger Stand:
 - ◆ Projekt-Hosting bei ***sourceforge.net***
 - ◆ Das 3. Mal im Schwerpunktpraktikum
 - ◆ Bereits 2 Diplomarbeiten zum Thema





Matrix4.plot und Eclipse

- Realisierung der Dotplot-Applikation als Eclipse-Feature (Plugin)
- Eclipse bietet:
 - ◆ das „Standard Widget Toolset“ (SWT) für GUI
 - ◆ Einbettung des Features als Eclipse-Perspektive
 - ◆ Konfigurationsmanagement usw.
- Rich-Client:
 - ◆ Standalone Applikation
 - ◆ Nur grundlegender Kern von Eclipse
 - ◆ ca. 90% Ressourcenersparnis



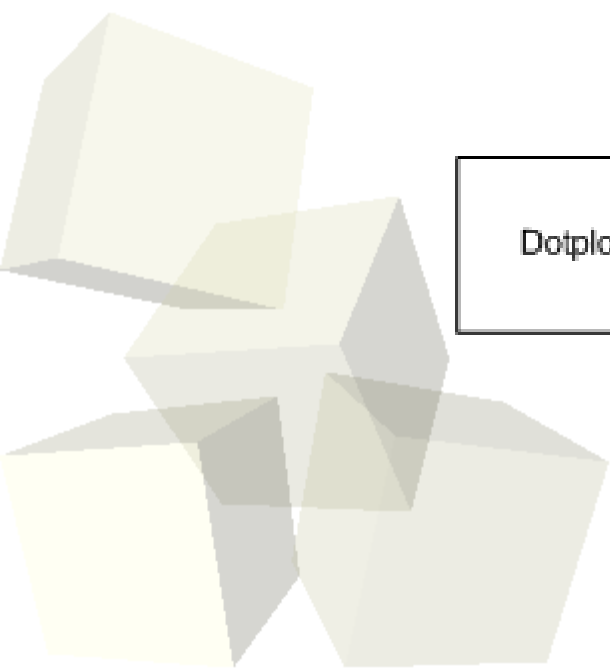
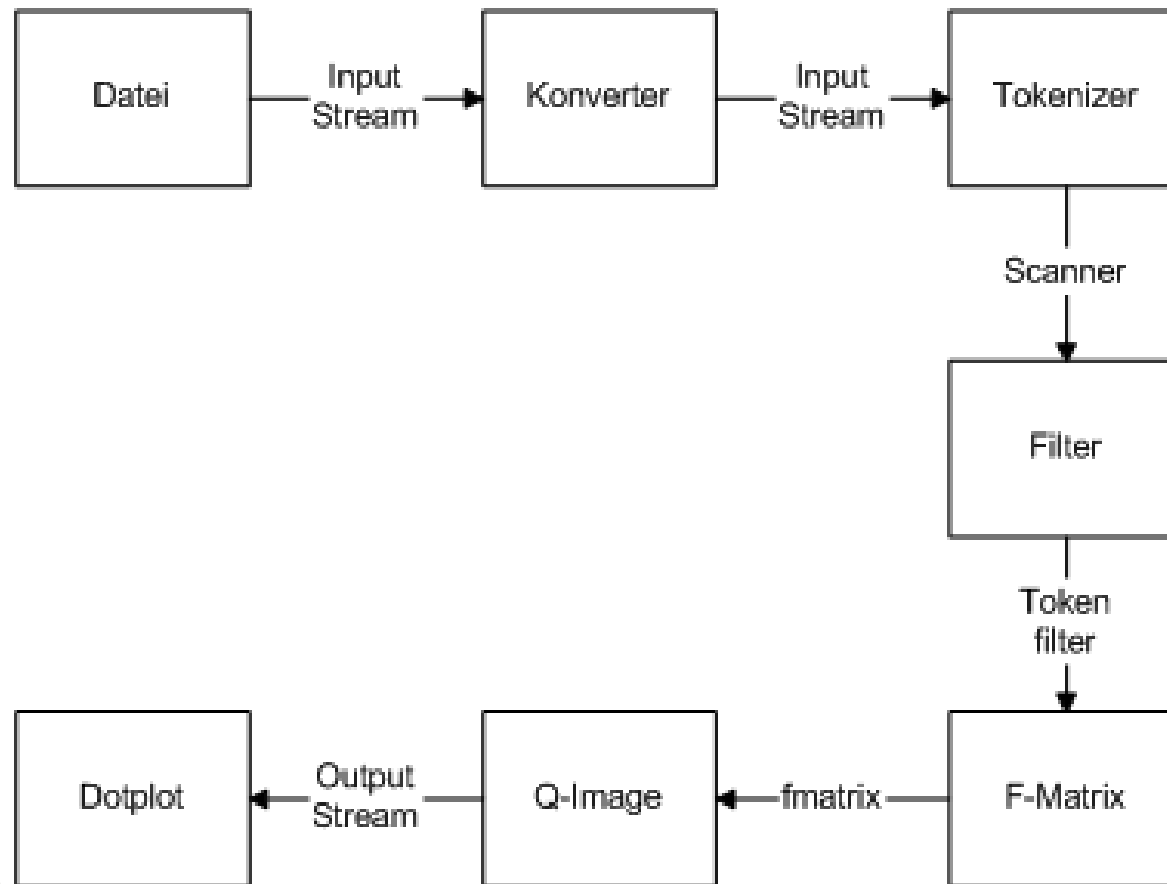
Matrix4.plot: Funktionsweise

- Tokenizer
 - ◆ Liest Daten aus einer Datei
 - ◆ Bildet „**Tokens**“ z.B.: mit Hilfe eines Scanners
- Filter: Benutzerdefinierte Filterung von Tokens
- F-Matrix:
 - ◆ Aufspannen in einer Matrix aus Fließkommazahlen.
 - ◆ Treffer und Gewichtung durch Fließkommawert
- Q-Image
 - ◆ Erzeugung des Bildes
 - ◆ Gewichtung erzeugt verschiedene Farben



Matrix4.plot: Funktionsweise (2)

■ Pipes & Filters



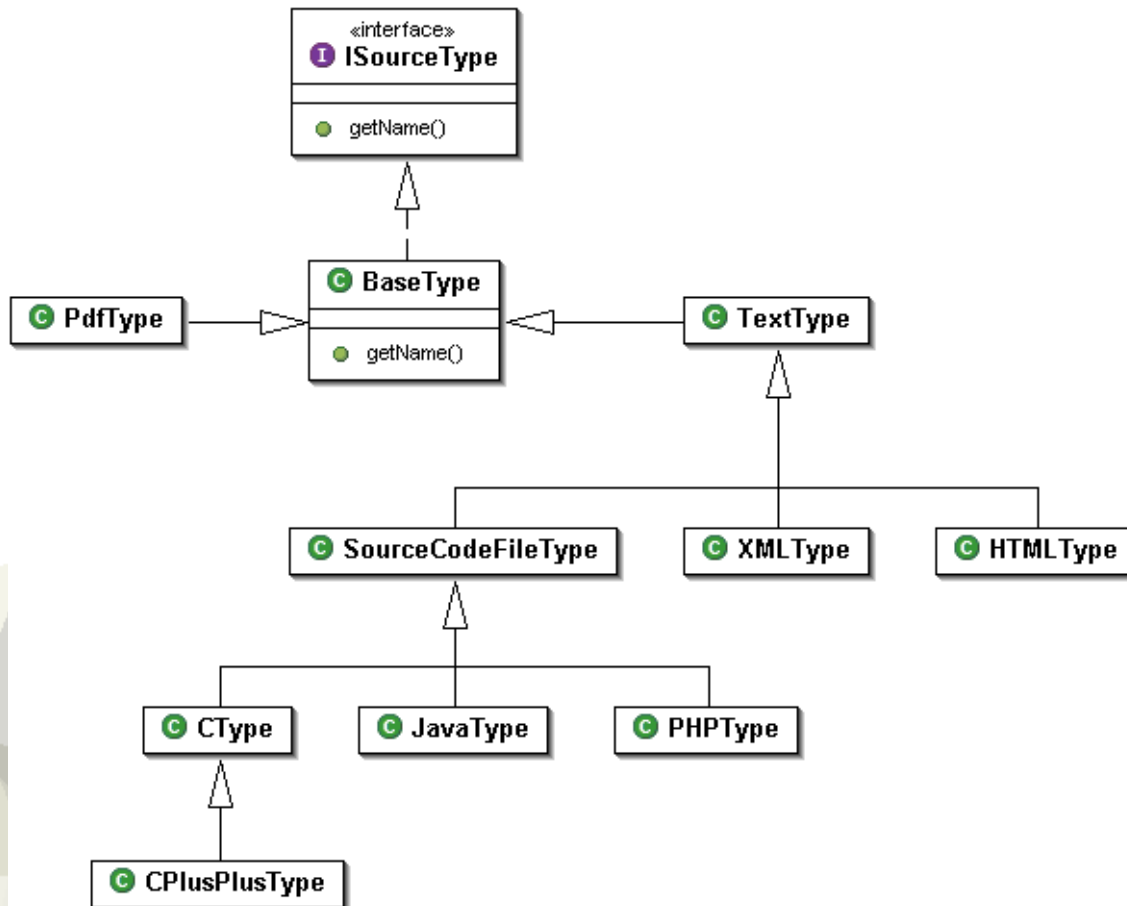


Konzeptionelle Änderungen

- Datenquelle für einen Dotplot
 - ◆ Nicht nur Datei als Input
 - ◆ Auch Daten aus Internet oder Datenbank
- Idee: Plotquelle
 - ◆ Name
 - ◆ Grösse
 - ◆ URL, dh.: lokalisierbar
 - ◆ Lesbar
 - ◆ Typ

Konzeptionelle Änderungen (2)

- Typisierung:
 - ♦ Plotquelle soll Typ haben
 - ♦ Typgerechte Verarbeitung





Konzeptionelle Änderungen (3)

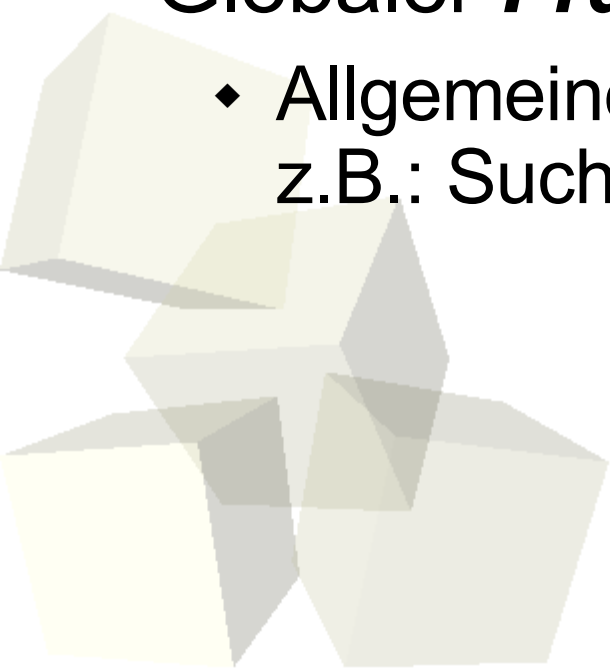
■ Konfigurationsmanagement

- ◆ Idee: Speicherung von Konfigurationsdaten beim Beenden der Applikation
- ◆ Realisierung durch Eclipse-Konfiguration möglich
- ◆ Problem: breite Abhängigkeit von der Eclipse-Plattform
- ◆ Lösung: Eigene Konfiguration, die intern auf Eclipse zugreift.
- ◆ Bei Plattformwechsel muss nur die eigene Konfiguration angepasst werden.



Service-Framework

- Soll grundlegende Funktionsweise von Matrix4.plot ermöglichen
- Pipes & Filters:
 - ◆ Hintereinanderschalten von Services zu einer Servicekette.
- Globaler ***FrameworkContext***
 - ◆ Allgemeine Dienste
z.B.: Suchen von Services in der Service-Registry





Services

- Was ist ein Service?
 - ◆ Eine gekapselte Ausführungseinheit
 - ◆ Persistentes Objekt
 - ◆ Ist registriert in der globalen Service-Registry mit eindeutiger ID
- Was macht ein Service?
 - ◆ Bekommt Input in Form eines **WorkingContext's**
 - ◆ Erzeugt Output in Form eines **ResultContext's**
- Wie arbeitet ein Service?
 - ◆ Erzeugt eine **Task** zur Erfüllung der Aufgabe
 - ◆ Instruiert die **Task** mit den Eingabedaten



Tasks

- Was ist/macht eine Task?
 - ♦ Erfüllt die Aufgaben eines **Services**
 - ♦ Besteht aus einzelnen **TaskParts**
- Wie arbeitet eine Task?
 - ♦ Ein **TaskProcessor** bestimmt wie und in welcher Reihenfolge die **TaskParts** bearbeitet werden.
 - ♦ Mögliche Parallelisierung von **TaskParts**
 - ♦ Ein **TaskMarshaller** erzeugt aus den Einzelergebnissen der **TaskParts** das Gesamtergebnis





Hotspots

- Was ist/macht ein Hotspot?
 - ♦ Ermöglicht die Erweiterung eines **Services**
 - ♦ Objekte mit neuer Funktionalität können in den **Service** integriert werden
 - ♦ Die Funktionalität wird durch Objekte des Typs **Extention** übergeben

- Was ist/macht eine Extention?
 - ♦ Enthält einen **ExtentionActivator** zum Starten der neuen Funktionalität
 - ♦ Enthält eine Klasse mit beliebig vielen Parametern
 - ♦ Die Klasse bietet die neue Funktionalität



- Was ist/macht ein Job?
 - ♦ Starten einer Aktion oder eines Ablaufs innerhalb des Service-Framework
 - ♦ z.B.: Aktivieren der Funktionalität eines **Services** oder Verkettung von mehreren **Services**
- Wie arbeitet ein Job?
 - ♦ Pipes & Filters
 - ♦ Ein **Service** ist ein Filter, mit Input und Output
 - ♦ Ein **Job** verknüpft den Output des einen Services mit dem Input des anderen Services.



Plugin-Framework

- Erweiterung des ***Service-Framework***
- Plugins werden zur Startzeit in das System integriert
 - ◆ Entwicklung von zusätzlicher Funktionalität möglich
- PluginContext
 - ◆ Erweiterung des ***FrameworkContexts***
 - ◆ ***PluginRegistry***
 - ◆ Plugin-Verzeichnis, von dem die Plugins geladen werden
 - Versionierung von Plugins



Plugins

- Ein Plugin kann **Services**, **Extentions** und **Jobs** enthalten und kapseln
- Wird in Form eines **jar**-Archives zur Verfügung gestellt
 - ◆ Klassen für die Funktionalität des Plugins
 - Services, Extentions und/oder Jobs
 - ◆ XML-Dokument zur Konfiguration und Beschreibung des Plugins (*dotplotplugin.xml*)
 - Version des Plugins
 - Was für eine Art von Funktionalität enthält das Plugin?
 - Welche Klassen repräsentieren die Bestandteile?
 - Welche Abhängigkeiten zwischen den Plugins

Beispiel: dotplotplugin.xml

```
1  <?xml version="1.0"?>
2  <Dotplotplugin
3      id="org.dotplot.examples"
4      name="Examples"
5      version="1.0"
6      provider="FH Giessen-Friedberg (www.fh-giessen.de)"
7      info="Examples of extention development for Matrix4.plot.">
8
9      <Dependency plugin="org.dotplot.core.Standard" version="1.0"/>
10
11     <Service id="org.dotplot.standard.Filter">
12
13         <Extention hotspot="org.dotplot.standard.Filter.newFilter"
14             class="org.dotplot.examples.FourGrammFilter.FourGrammFilter">
15             <Parameter name="name" value="org.dotplot.filter.4GrammFilter"/>
16             <Parameter name="ui" value="org.dotplot.examples.FourGrammFilter.FourGrammFilterUI"/>
17         </Extention>
18
19     </Service>
20
21     <Service id="org.dotplot.standard.EclipseUI"
22         class="org.dotplot.eclipse.EclipseUIService">
23
24         <Extention hotspot="org.dotplot.standard.EclipseUI.Menu"
25             class="org.dotplot.core.services.Structure">
26             <Parameter name="id" value="org.dotplot.ui.menue.Jobs" />
27             <Parameter name="name" value="Jobs"/>
28         </Extention>
29
30     ...
31
32     <Job
33         id="org.dotplot.jobs.TestJob"
34         class="org.dotplot.examples.TestJob" />
35 </Dotplotplugin>
```



Services für das Plugin-Framework

- 3 Services zum Laden, Initialisieren und Integrieren von Plugins
 - ♦ ***PluginLoadingService***
 - Laden der Plugin-Klassen
 - Auflösen der Abhängigkeiten
 - ♦ ***PluginIntegrationService***
 - Integration aller Plugins in den PluginContext (Registry)
 - ♦ ***InitializerService***
 - Initialisierung der neu integrierten Services
 - 2 Hotspots: ***Startup*** und ***Shutdown***
 - Benutzerdefinierte Funktionen beim Aufstarten oder Herunterfahren des Systems



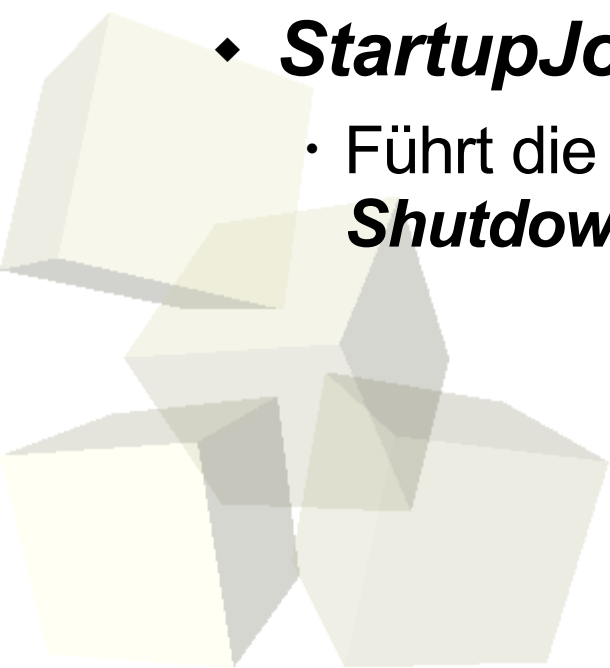
Services/Plugins in Matrix4.plot

- Unsere Dotplot-Applikation ***Matrix4.plot*** soll nun mit dem Service-/Plugin-Framework zusammenarbeiten
- Core-Plugin (Kern von Matrix4.plot)
 - ◆ Ist fest im Framework implementiert
 - ◆ Wird automatisch geladen
 - ◆ Enthält die 3 vorgestellten Services des Plugin-Frameworks:
 - ***PluginLoadingService***
 - ***PluginIntegrationService***
 - ***InitializerService***



Core-Plugin (2)

- Core-Plugin (Kern von Matrix4.plot)
 - ◆ ...
 - ◆ Enthält 3 Systemjobs:
 - ◆ ***PluginLoadingJob***:
 - bildet aus den 3 ***Plugin-Framework-Services*** eine Servicekette
 - Lädt die übrigen Plugins
 - ◆ ***StartupJob* / *ShutdownJob***:
 - Führt die Erweiterungen der beiden Hotspots, ***StartUp*** und ***Shutdown***, von ***InitializerService*** aus



■ *DotplotContext*

- ◆ Dotplot-spezifische Erweiterung des *PluginContexts* bzw. *FrameworkContexts*
- ◆ Zentrale Anlaufstelle für die gesamte Applikation
- ◆ Enthält Registry für *Services*, *Plugins*, *Jobs* usw.

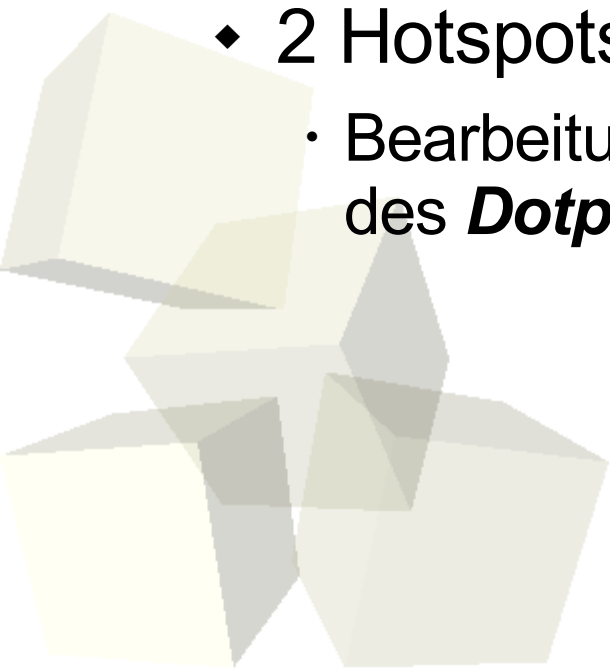
■ Dotplot-spezifische Erweiterungen des *DotplotContexts*

- ◆ ConfigurationRegistry:
 - Abspeichern der Konfiguration eines Services in der Eclipse-Konfiguration
- ◆ TypeRegistry
- ◆ TypeBindingRegistry



■ **DotplotService**

- ◆ Oberklasse von (fast) jedem Service innerhalb der Dotplot-Applikation
- ◆ Verallgemeinerung von sinnvoller Funktionalität für jeden **DotplotService**
- ◆ Konfigurations-Objekt zum Abspeichern von Konfigurationsdaten
- ◆ 2 Hotspots **Before** und **After**:
 - Bearbeitung von Funktionen **vor** bzw. **nach** der Ausführung des **DotplotService**

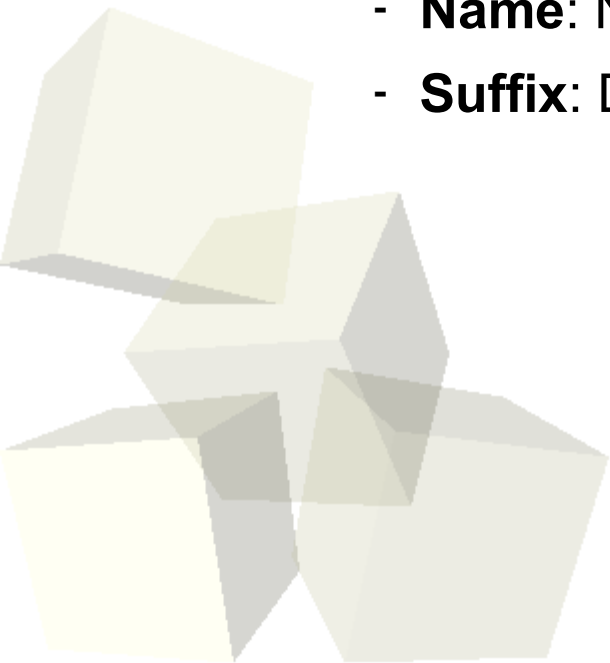




Services von Dotplot: Converter

■ Service: **Converter**

- ♦ Konvertiert Quelldaten von einem Typ in den anderen (z.B.: Pdf-Textdokument in ASCII-Text)
- ♦ 2 Hotspots:
 - **newConverter**: Einfügen eines neuen Konverters
 - **newType**: Einfügen eines neuen Typs und Bindung an eine Dateiendung. 2 Parameter:
 - **Name**: Name des Typs
 - **Suffix**: Dateiendung

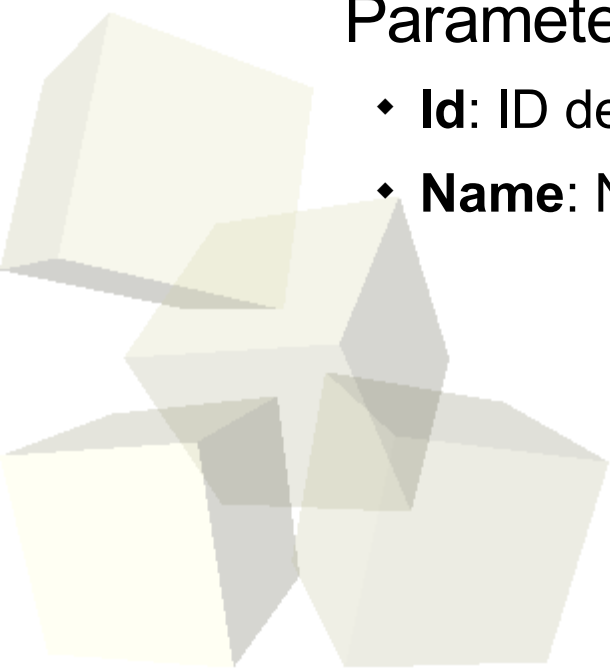




Services von Dotplot: Tokenizer

■ Service: *Tokenizer*

- ♦ Verwandelt eine Liste von Plotquellen in einen „Strom von Tokens“.
- ♦ Zuweisung des **typgerechten** Tokenizers (z.B.: Scanner)
- ♦ 1 Hotspot:
 - ***newTokenizer***: Einfügen eines neuen Tokenizers.
Parameter:
 - ♦ **Id**: ID des Tokenizers
 - ♦ **Name**: Name des Tokenizers



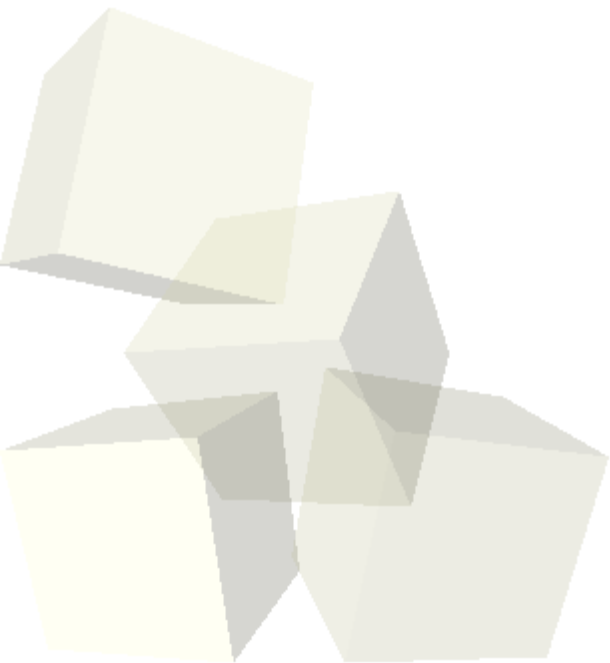
■ Service: *Filter*

- ♦ Schaltet mehrere Filter hintereinander
- ♦ Filtert einen „Strom von Tokens“
- ♦ Anhand der Einstellungen eines Filters wird entschieden, welcher Filter aktiv ist und wie er arbeitet.
- ♦ 1 Hotspot:
 - ***newFilter***: Einfügen eines neuen Filters. Parameter:
 - ♦ **Name**: Name des Filters
 - ♦ **Ui**: Eine Klasse, mit der die Einstellungen des Filters vorgenommen werden können.



Services: F-Matrix/Q-Image

- Service: ***F-Matrix***
 - ♦ Dieser Service erzeugt aus dem Strom von Tokens die F-Matrix
- Service: ***Q-Image***:
 - ♦ Generierung des fertigen Bildes aus der F-Matrix





Services von Dotplot: GUI

■ Service: **GUI**

- ♦ Dieser Service ermöglicht das Einfügen von „Konfigurationsansichten“ und „Menü-Einträge“ für die Dotplot-Applikation. 3 Hotspots:
 - **View**: Einfügen einer neuen Konfigurationsansicht:
 - ♦ **Id**: ID der Konfigurationsansicht
 - **Menu**: Einfügen eines Menü- bzw. Submenü-Eintrags:
 - ♦ **Id**: ID des Menüs
 - ♦ **Name**: Name des Menüs in der Menüleiste
 - ♦ **Menu**: ID des Elternmenüs
 - **Entry**: Einfügen eines Eintrags in einem Menü zum Aktivieren eines Jobs
 - ♦ **Name**: Name des Eintrags in der Menüleiste
 - ♦ **Menu**: ID des Elternmenüs
 - ♦ **Job**: Klassenname des auszuführenden Jobs



Fazit und Quellen

■ Fazit:

- ♦ Das neue Service-Framework in Verbindung mit dem Plugin-Framework bietet:
 - Modularer Aufbau
 - Nachträgliche Entwicklung von Funktionalität
 - Erweiterung von Framework-Komponenten über Hotspots

■ Quellen

- ♦ Hauptquelle zum Service-Framework:
„**Mustergetriebene Anwendungsentwicklung**“ von Christian Gerhardt (Diplomarbeit)
- ♦ *<http://sourceforge.net/projects/dotplot>*